(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 173 905
A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 85110350.7

(22) Date of filing: 20.08.85

(51) Int. Cl.⁴: G 06 F 12/06

(30) Priority: 07.09.84 US 648541

(43) Date of publication of application:
12.03.86 Bulletin 86/11

(84) Designated Contracting States:
DE FR GB NL

(71) Applicant: TEKTRONIX, INC.
Tektronix Industrial Park D/S Y3-121 4900 S.W. Griffith Drive P.O. Box 500
Beaverton Oregon 97077(US)

(72) Inventor: Sullivan, James P.
3644 S.E. Gladstone Street
Portland Oregon 97202(US)

(74) Representative: Strasse, Joachim, Dipl.-Ing. et al,
Strasse und Stoffregen European Patent Attorneys
Zweibrückenstrasse 17
D-8000 München 2(DE)

(54) Dynamic address assignment system.

(57) For a computer system having peripheral devices coupled to a common bus through interface devices transmitting and receiving messages containing an address code matching a stored address code, a dynamic address assignment system stores a unique address code in each interface device following system startup. On system startup each interface device stores a type number and an adjustable serial number, type numbers for peripheral devices of the same type being identical while serial numbers for all peripheral devices of the same type are adjusted to different values. A master controller transmits to all peripheral devices a series of universally addressed count commands. Each interface device counts the count commands and, when the count reaches a poll number determined by the unique combination of stored type and serial numbers, requests and obtains a unique address code from the host computer. The interface device thereafter stores and uses the unique address code in transmitting and receiving messages on the bus.

EP 0 173 905 A2

read only memory (ROM) contained in each device. A communication controller in each peripheral device is then programmed to respond only to messages containing addresses matching the number stored in ROM or set by the switches. One disadvantage of using a ROM in each peripheral device is that the ROM must be different for each device. If, for instance, two keyboards are connected to the same bus, the ROM in one keyboard must be altered.

Use of switches in each peripheral device makes it easier for a user to give each device a unique address. However a large number of switches must be used if the address uses several bits to accommodate a large number of peripheral devices. Requiring a user to set a large number of switches accurately presents an opportunity for human error.

It would be advantageous if peripheral devices could be added to a serial bus without the need for changing ROMs in any peripheral device or without the need for setting the entire device address in switches in the peripheral devices.

Summary of the Invention

The present invention is a method and apparatus for dynamically addressing a plurality of peripheral devices communicating with a central or "host" computer over a single serial data bus. Each peripheral device contains a remote universal peripheral interface device (PID) and a set of four switches. The host computer is connected to a master controller through a bidirectional parallel port. The master controller and all PIDs each have a serial port connected to the serial data bus. Thus all information passing between the host computer and a peripheral device passes through

number and the switch position code. On receipt of a "REQUEST SID" message, the master PID stops transmitting CONFIG commands, and transmits the peripheral type number and switch position code to the host computer. The host computer then determines a "short identification number" (SID number) to be used as an address code for the peripheral and then issues an "ASSIGN SID" command. The ASSIGN SID command is transmitted over the serial bus by the master controller along with the type number and serial number received in the REQUEST SID message and the SID number.

The PID sending the REQUEST SID message is programmed to respond to an ASSIGN SID command containing its own peripheral type number and serial number and the ASSIGN SID command causes the PID to store the SID number in an internal address register. Thereafter the PID will respond to any incoming message transmitted under that particular SID number and will include that SID number in every outgoing message. The SID number thus becomes that peripheral's unique address until the system is restarted.

After transmitting the ASSIGN SID command, the master controller resumes issuing CONFIG commands until it reaches a preset count limit equal to highest possible poll number associated with any PID. On reaching the count limit the configuration cycle is complete, each peripheral having requested and obtained a unique SID number address. Subsequent communications between host and peripherals may then be transmitted using SID number address codes.

The present invention thus allows the host computer to establish peripheral addresses dynamically after system start up. All peripherals of the same type contain the same ROM and no manual adjustment to peripheral hardware is required to distinguish peri-

peripheral connected to a serial bus.

The invention resides in the combination, construction, arrangement and disposition of the various component parts and elements incorporated in the present invention. The present invention will be better understood and objects and important features other than those specifically enumerated above will become apparent when consideration is given to the following details and description, which when taken in conjunction with the annexed drawings describe, disclose, illustrate, and show a preferred embodiment or modification of the present invention and what is presently considered and believed to be the best mode of practicing the principles thereof.

Description of the Drawings

FIG. 1 is a block diagram depicting a system incorporating the preferred embodiment of the present invention,

FIG. 2 is a flowchart depicting the configuration mode operation of the master interface controller of FIG. 1, and

FIG. 3 is a flowchart depicting the configuration mode operation of the peripheral interface device of FIG. 1.

Detailed Description of the Preferred Embodiment

FIG. 1 is a block diagram of a modular input system (MIS) 10 incorporating the present invention for providing a communication link between host computer 20 and one or more peripheral devices 30 such as key-

another peripheral device of the same type. Each PID 50 interrogates the switch positions and places a number (from 0 to 15 decimal) representing the particular switch settings in serial number register 54 which may be contained in random access memory (RAM) 48 accessed by processor 56.

Each PID 50 contains a type number count register 58, a serial number count register 62, address register 80 and a communication buffer 64, all of which may be contained in RAM 48. Processor 56 uses communication buffer 64 to temporarily store data transmitted over bus 46. The registers are explained below.

Master interface controller 40 also has a processor 70 controlled by a program contained in ROM 72, a communication buffer 74 for temporary storage of data transmitted over bus 46, and poll count register 76. Poll count register 76 and buffer 74 may be contained in RAM 78.

FIG. 2 is a flow chart of a routine contained in ROM 72 for controlling the operation of master interface controller 40 during system configuration following system start up. System configuration begins when controller 40 receives a "CONFIG" command from host computer 20 over bus 22. This occurs during system start up or reset. The purpose of the configuration routine is to allow host computer 20 to assign addresses to each peripheral device 30.

Referring to FIGS. 1 and 2, upon receipt of a CONFIG command from host computer 20 initiating start (block 101 of FIG. 2) of the configuration routine, system controller 40 (in block 102) sets poll count register 76 to zero, transmits a CONFIG command out on bus 46 to all PIDs 50 using universal address "FE" and starts a "TIMEOUT" timer. In blocks 104 and 105 controller 40 waits for for the duration of the TIMEOUT timer

ler 40, incrementing type number count register 58 once for every 16 CONFIG commands. Serial number count register is incremented once for every CONFIG command but is reset to zero after every sixteen counts. When the count stored in type number count register 58 matches the peripheral type number stored in ROM 58, and when the number stored in serial number count register 62 matches the serial number stored in serial number count register 54, PID 50 transmits a REQUEST SID message to master controller 40 and receives an ASSIGN SID command in return containing the SID number as described above for use as its address code. Since the combination of serial number and type number is unique for each PID, each PID receives a unique SID number. Each PID 50 stores its SID number in address register 80.

FIG. 3 is a flow chart of a program contained in ROM 52 for controlling the operation of PID 50 during system configuration. Turning now to FIGS. 1 and 3, the PID configuration routine starts at block 201 on system power up. In block 202, each PID 30 loads the hexadecimal number FE into address register 80. Each PID responds to information transmitted over bus 46 when preceeded by the number contained in register 80. Thus after system start-up, every PID 50 responds to messages sent to address FE. Also in block 202, each PID sets a "TYPE MATCH" marker to a logical false condition. This marker is used to indicate whether the CONFIG command count stored in type number count register 58 has reached the type number stored in ROM 50. A logical false condition indicates that a match has not yet occured.

With address FE in register 80 every PID 50 then waits in block 203 for the first CONFIG command transmitted by master controller 40 over bus 46. On receipt of the first CONFIG command, each PID zeros the con-

52, block 207 directs PID 50 to block 213 where the type match marker is set to logical true. Thereafter, in 214, PID 50 compares the contents of serial number count register 62 with the peripheral serial number stored in register 54. If there is a match, then PID 50 transmits a REQUEST SID message to controller 40 according to block 215. If there is no match, or after a REQUEST SID message is sent, the serial number count in register 62 is incremented in block 208 and PID 50 is directed back again to the blocks 205 and 216 loop to wait for another CONFIG command or an ASSIGN SID command. On receipt of subsequent CONFIG commands, block 206 will direct PID 50 directly to block 214, bypassing block 213 since the type match marker is already true.

It should be noted that only one PID 50 will send a REQUEST SID command at a time because the combination of type number and serial number is unique for each PID 50. The type number stored in ROM 50 is unique for each peripheral 30 type while the serial number in register 54 is uniquely set to distinguish between peripherals of the same type.

The REQUEST SID message, sent to controller 40 in block 215, contains the type and serial number of the initiating PID 50. Controller 40 passes this number to host computer 20. Host computer 20 may use this type number to identify the peripheral type so that it can later use the proper interface routines when communicating with the peripheral. Host computer 20 determines an appropriate SID number to use for the peripheral address and transmits it back to the PID 50 via an ASSIGN command. The ASSIGN command contains the type and serial number of the PID 50 making the SID request.

All PIDs 50 receive the ASSIGN command but only the requesting PID 50 stores the SID number transmitted in the ASSIGN command in its address register 80. In

unique number for storage in serial number register 54 of FIG. 1.

The following is a simplified code listing implementing the flow chart of FIG. 2.

```
10  POLL = 0
20  SEND CONFIG TO ADDRESS FE
30  START TIMEOUT
40  IF SID REQUEST RECEIVED THEN GOTO 60
50  IF TIMEOUT LIMIT THEN GOTO 100
60  START TIMER
65  GET SID FROM HOST
70  SEND ASSIGN SID TO ADDRESS FE
80  IF ACKNOWLEDGE RECEIVED GOTO 100
85  IF TIMER LIMIT THEN GOTO 90 ELSE GOTO 80
90  SEND ACKF TO HOST
100 POLL = POLL + 1
110 IF POLL <> 4096 THEN GOTO 20
120 END
```

The following is a simplified code listing implementing the flow chart of FIG. 3.20 .

```
05  ADDRESS = FE
10  TMATCH = 0
20  IF CONFIG NOT RECEIVED THEN GOTO 225
30  TCOUNT = 0
40  SCOUNT = 0
50  IF CONFIG RECEIVED THEN GOTO 90
60  IF ASSIGN NOT RECEIVED THEN GOTO 50
70  IF TYPE = TNUMBER AND SERIAL = SNUMBER THEN
    ADDRESS = SID ELSE GOTO 50
80  END
90  IF TMATCH=1 THEN 120
100 IF TCOUNT <> TYPE THEN 130
```

Claims:

1. For a computer system of the type having a plurality of peripheral devices communicating with a host computer through a common serial bus, a method of uniquely identifying each peripheral device comprising:

storing digital data representing a type number and an adjustable serial number in each peripheral device, type numbers stored by all peripheral devices of the same type being identical, serial numbers stored by peripheral devices of the same type being adjusted to different values such that the combination of stored type and peripheral numbers is unique for each peripheral device.

2. A method as in claim 1 wherein a type number is stored in a read only memory accessable by an associated peripheral device.

3. A method as in claim 1 wherein a serial number is adjusted by changing the position of at least one switch.

4. A method as in claim 1 wherein a serial number is adjusted by selective placement of at least one jumper.

5. For a computer system of the type having a plurality of peripheral devices communicating with a host computer through a common serial bus, each peripheral device being coupled to the bus by an associated interface device for transmitting and receiving messages accompanied by an address code matching a stored

causing each interface device to count the count commands occuring on the bus and to obtain from the host computer a unique address code when the count reaches a poll number uniquely determined by a combination of the associated type and serial numbers.

7. For a computer system of the type having a plurality of peripheral devices communicating with a host computer through a common serial bus, each peripheral device being coupled to the bus by an associated interface device for transmitting and receiving messages accompanied by an address code matching a stored address code, a method for dynamically assigning and storing a unique address code in each interface device following system startup comprising the following steps:

storing a type number and an adjustable serial number in each interface device, type numbers associated with all peripheral devices of the same type being identical, serial numbers associated with peripheral devices of the same type being adjusted to different values;

setting the stored address in each interface device to a univeral address code upon system start-up;

transmitting a series of universally addressed count commands on the bus;

causing each interface device to count the count commands occuring on the bus and to transmit an address request message on the bus containing the associated type and serial numbers when the count reaches a poll number uniquely determined by a combination of the

dressed count commands on the bus;

causing each interface device to count the count commands occuring on the bus and to transmit an address request message on the bus containing the associated type and serial numbers when the count reaches a poll number uniquely determined by a combination of the associated type and serial numbers;

obtaining a unique address code from the host computer based on the type and serial number contained in the address request message;

transmitting on the bus a universally addressed addressing command containing the type number, the serial number and the unique address code; and

replacing the universal address code stored in each interface device with the unique address code contained in the addressing command when the addressing command contains the stored type and serial numbers, the interface device thereafter using the unique address code in transmitting and receiving messages on the bus.

9. A method as in claim 8 wherein the number of count commands transmitted equals at least the highest possible poll number associated with any interface device.

10. For a computer system of the type having a plurality of peripheral devices communicating with a host computer through a common serial bus, each peripheral device being coupled to the bus by an associated interface device for transmitting and receiving mes-

stored in each interface device with the unique address code contained in the addressing command when the addressing command contains the stored type and serial numbers, the interface device thereafter using the unique address code in transmitting and receiving messages on the bus.

11. An apparatus as in claim 10 further comprising a read only memory associated with each interface device for storing the peripheral type number.

12. An apparatus as in claim 10 further comprising at least one switch for adjusting a peripheral serial number.

13. An apparatus as in claim 10 comprising at least one cuttable jumper for adjusting the peripheral serial number.

0173905

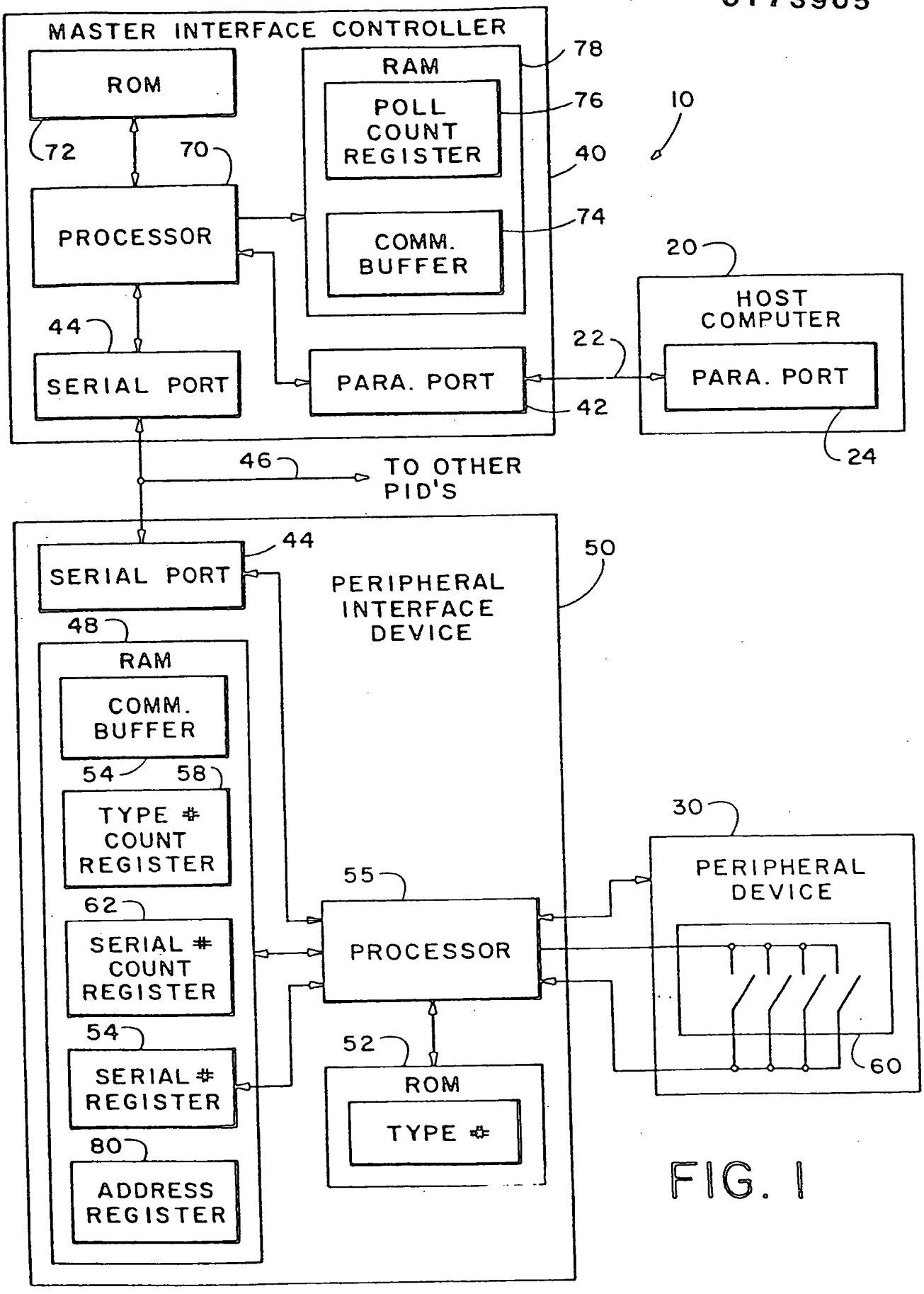

FIG. 1

0173905

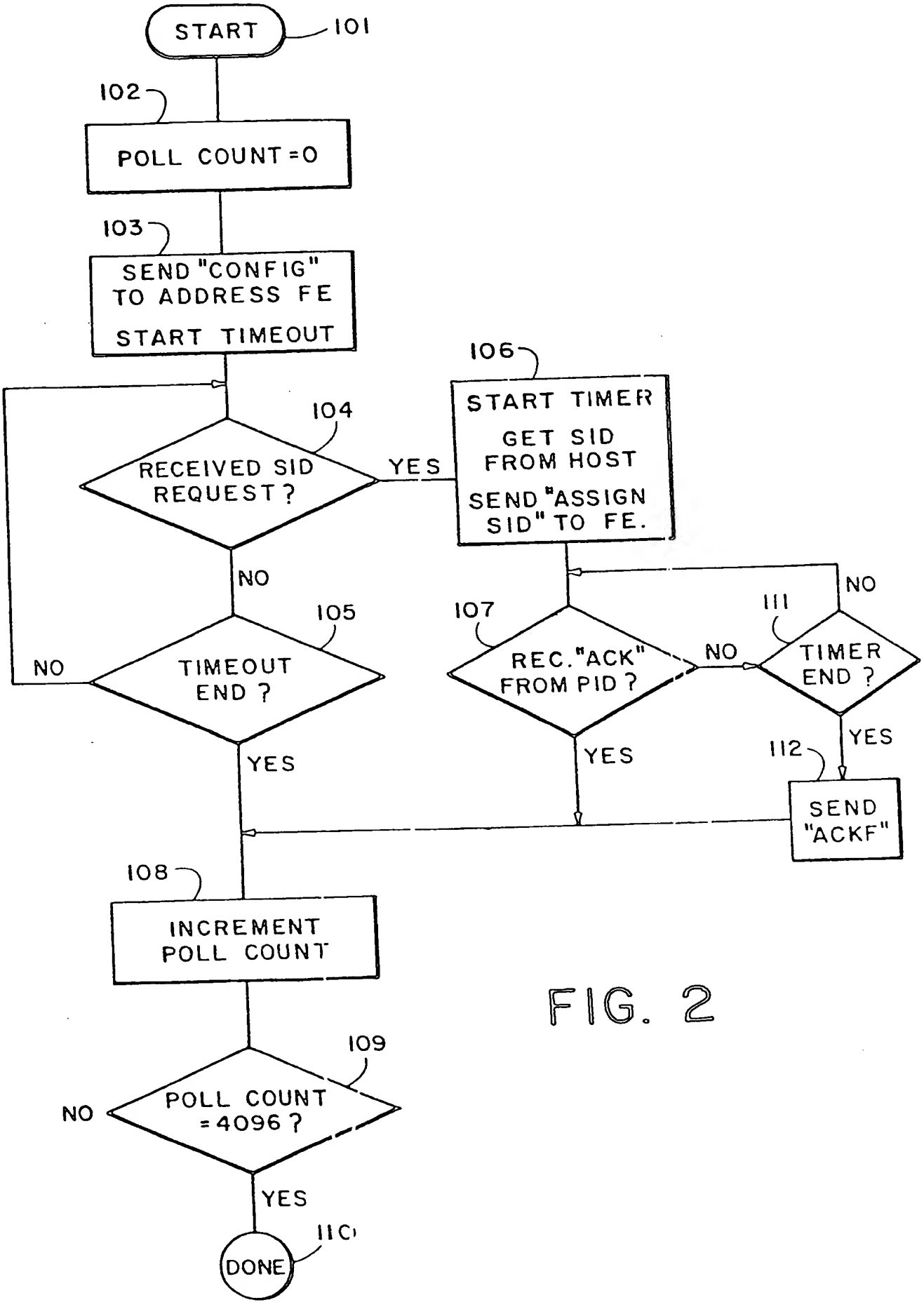

FIG. 2

0173905

FIG. 3